MECS 560-2 Dynamic Optimization

Online Learning

Dominic Anene

June 30, 2018

Summary

Online learning is the process of making predictions in which data becomes available in a sequential order and is used to update our best predictor for future data at each step. As opposed to *offline learning* which generate the best predictor by learning on the entire training data set at once.

Online learning is performed in a sequence of consecutive periods. At round t the learner is given a question $\mathbf{x}_t$ encoded with data, taken from a *input space (or instance domain)* $\mathcal{X}$, and is required to provide an answer to this question, denoted by p_t. The learner chooses the prediction value p_t in the *prediction space* S, on the basis of the previous $t-1$ observations. After predicting an answer, the correct answer y_t is revealed from the *outcome space* $\mathcal{Y}$, and the learner suffers a loss according to loss function $l(p_t, y_t)$. In many cases the prediction value p_t is in $\mathcal{Y} = S$ but sometimes it's convenient to let the learner pick a prediction from a larger space, S, where $\mathcal{Y} \subset S$.

The essence of the online learning is the view that the sequence of answer, $\{y_t\}_{t\geq 1}$, are the product of some unknown and unspecified mechanism of nature, which can be deterministic, stochastic, or adversarially adaptive to the learners behavior. Avoiding assumptions on the way the sequence to be predicted, $\{y_t\}_{t\geq 1}$, are generated we need a baseline for performance. In the basic model, the performance of the forecaster at period T is measured by the loss accumulated up until that period, scored by our loss function l.

For another type of baseline *regret*, we introduce a hypothesis class, also called a class of reference forecasters, also called a set of experts, U. The elements of U are functions $u : \mathcal{X} \to \mathcal{Y}$. These 'experts', $u \in U$, make their predictions available to the learner every period. The forecaster then makes predictions that depend on the 'experts' advice. [1] The difference between the learners accumulated loss at period T and that of an arbitrary expert is called *regret*. It measures how much the forecaster regrets, in hindsight, of not having followed the advice of expert u.

$$\text{Regret}_T(u) = \sum_{t=1}^{T} l(p_t, y_t) - \sum_{t=1}^{T} l(u(x_t), y_t) \tag{1}$$

The learners goal is to have the lowest possible regret relative to the set of experts U, so we measure accumulated loss vs the accumulated loss of the best expert, and we want to minimize this bad boy:

$$\text{Regret}_T(U) = \sum_{t=1}^{T} l(p_t, y_t) - \min_{u\in U} \sum_{t=1}^{T} l(u(x_t), y_t) = max_{u\in U}\text{Regret}_T(h). \tag{2}$$

Aka minimize maximum regret. It is often the case that we want regret, $\text{Regret}_T(U)$, to grow *sub-linearly* [2] with the number of periods, T. Which means that the average regret $\frac{1}{T}\text{Regret}_T(U) \to 0$ uniformly over outcome sequences $\{y_n\}$, and experts $u \in U$:

$$\frac{1}{T}\text{Regret}_T(U) = \frac{1}{T}[\sum_{t=1}^{T} l(p_t, y_t) - \min_{u\in U} \sum_{t=1}^{T} l(u(x_t), y_t)] \to 0, \text{ as } T \to \infty, \tag{3}$$

equivalently

$$max_{u\in U}\text{Regret}_T(u) = o(n)$$

Another type of baseline measure for performance is generated under the *realizable case assumption*, in which we assume that the true answer is generated by an 'expert' within U:

$$(\exists u^* \in U)(\forall t \in T)[u^*(\mathbf{x}_t) = y_t] \tag{4}$$

[1] In Manski 480-1 we encountered the expert problem in the 'Plannning under ambiguity' section. In this context the learner had to choose an action from a choice set C, to maximize an objective function $f(.) : C \to \mathbb{R}$ where $f \in F$ the set of functions. The learner knows the choice set C and that the objective function f is an element of F. In this context F is the set of experts. Recall that in Manski's class in addition to looking at regret we looked at maximin and bayes descision, these are simply 'baseline' measures of performance.

[2] If we can design an algorithm that is sub-linear this is called 'Hannan consistency'

This assumption just means that there is a expert $u^* : \mathcal{X} \to \mathcal{Y}$ that is always correct (we just don't know who it is). The adversarial environment is still out to get the learner and can mess with the data, $\mathbf{x}_t$, and it can mess with every expert (not including expert u^*) making them give you phony predictions so that your prediction is off. But the adversary has one hand tied behind his back since he can't mess with the truth telling expert, so in time we can find out who the truther is and win the game. [3] For an online learning algorithm, A, denote $M_A(U)$ to be the *maximal number of mistakes algorithm A might make*. The baseline for performance under this type environment is the *mistake bound*, where we want to design an algorithm for which $M_A(U)$ is minimal.

1 Online Learning

1.1 Expert Problem

Prediction with Expert Advice

Parameters: decision space S, input space $\mathcal{X}$, outcome space $\mathcal{Y}$, loss function l, set of U of experts.

Suppose the set of experts is finite, $|U| < \infty$. Index the set of experts s.t $U = \{u_1(\mathbf{x}_t), \ldots, u_N(\mathbf{x}_t)\}$, where $u_i : \mathcal{X} \to \mathcal{Y}$. So at each round an expert gets data and makes a prediction based on the data.

There are $t = 1, 2, \ldots, T$ periods, where at period t the learner has access to a vector of information $\mathbf{x_t} \in \mathcal{X}$. He must decide which expert, $u \in U(\mathbf{x}_t)$, to choose. Thus the decision space is vector of expert recommendations, $S = U(\mathbf{x}_t)$, the learner chooses prediction (expert recommendations) $p_t \in S$. After the prediction, the true answer, $y_t \in \mathcal{Y}$, is revealed by environment and the learner receives a loss $l(p_t, y_t)$.

[a]

[a] Usually $\mathcal{Y} = [0, 1]^N$ or in classification problems $\mathcal{Y} = \{0, 1\}^N$.

This problem could be formulated as a decision maker facing the choice between buying and selling a stock with the assistance of N experts.

1.2 Expert Classification Problem under Realizability Assumption

Algorithm: An algorithm is a function $g_t : (\mathbf{x}_1, y_1) \times \cdots \times (\mathbf{x}_{t-1}, y_{t-1}) \to p_t$. Equivalent notation $g_t : \prod_{i=1}^{t=1}(\mathbf{x}_1, y_1)$ *Aka an Algorithm is a function that uses the past to make a prediction*

In classification problems we have the following primitives:

- Outcome space: $\mathcal{Y} = \{0, 1\}$
- Hypothesis class space: $\forall t \in T[U(\mathbf{x}_t) \subset \{0, 1\}^N]$
  - experts give binary $0 - 1$ recommendations
- Loss function: $l(p_t, y_t) = |p_t - y_t|$
  - $l(p_t, y_t)$ *is a* $0 - 1$ *loss function where* $l(p_t, y_t)$ *indicates if* $p_t = y_t$ *(correct prediction), or* $p_t \neq y_t$ *(incorrect prediction).*

Let us assume the *realizability assumption* (4). So there exists an expert $u^* \in U$ s.t u^* is always correct. Under this type of assumption our goal is to design an algorithm, A, that minimize the the maximum number of mistakes $M_A(U)$. In general we look algorithm that for each period $t \in T$ is consistent, meaning it maintains a set, V_t,

[3] In these types of games convergence rates should be pretty important, since even with a shit algorithm the learner will win eventually, but it could take a long time.

that is defined by the same rule each round, that's based on the sequence of past inputs and past observations up to time $t-1$, $\{(\mathbf{x}_j, y_j)\}_{j=1}^{t-1}$. The algorithm chooses experts from this set, V_t, each round.

Before we begin defining algorithms can you think of an algorithm, A, that minimize the maximum number of mistakes $M_A(U)$ under the *realizability assumption*? This algorithm that you think of has to think of a way to eliminate shit experts, eventually converging to the truther expert i^*. Implicitly this means the set ,V_t, of experts you choose from each round is monotonically decreasing in t converging to the truther $\{i^*\}$. Now instead of keeping track of remaining experts with the set V_t you can also choose some weighting scheme, aka gives experts weights each round to keep track. Keep in mind the enviornment that you are picking an algorithm under is adaptive adversarial, the adversary will do everything possible that algorithm is shit, if you change your algorithm the adversary will adapt with you and make sure your new algorithm is shit. The adversary has control over answers $\{\mathbf{y}_t\}_{t\geq 0}$, he has control over the information $\{\mathbf{x}_t\}_{t\geq 0}$, and through this control of information control of experts predictions (which depend on information $\{\mathbf{x}_t\}$ to make reccomendations). Therefore the adversary can make each and every experts you choose give you the wrong prediction (except the truther i^*). The adversary is deprave and a cheater, he can look at your prediction each round and even if you are correct he will tell you that you are wrong. The key to this problem is that the adversary does not have control over truther i^* who always gives the correct answer regardless of what the adversary does if you pick him/her, but ... you dont know who this truther is ... yet ...

Now that you have guessed lets analyze the simplest type of algorithm that we can employ in this type of environment under the *realizability assumption* called the **Consistent Algorithm**, otherwise called the 'shit algorithm'. Let U be the index set of experts and set $S = U$, and hence s_i is the identifier for expert i, $u_i(\mathbf{x}_t) \in U$ is expert i prediction at round t given information $\mathbf{x}_t$. Denote the truth telling experts index by i^*.

Consistent Algorithm

initialize: $V_1 = S$ (*start with every expert*)
for $t = 1, 2, \ldots$
receive $\mathbf{x}_t \in \mathcal{X}_t$ (*receive data* $\mathbf{x}_t$)
choose any $s_i \in V_t$ (*choose any expert in our restricted set of experts*)
predict $p_t = s_i$ (*choose expert i's prediction, implicitly* $p_t = s_i = u_i(\mathbf{x}_t)$)
receive answer $y_t = s_{i^*}$ (i^*, *gives us the truth,* y_t, *but remains unknown*)
update $V_{t+1} = \{i \in V_t : s_i = y_t\}$ *(eliminate experts who have failed us)*

Property of the Consistent Algorithm: For a finite set of experts $|U| = N$. The consistent algorithm guarantees the mistake bound $M_{\text{consistent}} \leq N - 1$.[a]

[a]This follows because if we make one mistake, then we eliminate the expert who gave us bad information from the set V_t. Therefore, after making M mistakes $|V_i| \leq N - M$, but
$\forall t \in T(d_i* \in V_t)$, aka the truth telling expert is always in V_t, and so $1 \leq |V_t|$. Combining both results we have that for all mistakes, M, we have $1 \leq N - M \iff M \leq N - 1$, since this holds for all mistakes M this holds for maximum, and so $M_{\text{consistent}} \leq N - 1$.

This algorithm sucks. It's convergence rate is pretty slow and number of mistakes is pretty bad. We need to figure out a smarter way to choose an expert $s_i \in V_t$. So instead picking *any ole'* expert, $s_i \in V_t$, let's be a bit smarter about our choice, and hence we introduce the majority algorithm.

Majority(Halving) Algorithm The premise of this algorithm is as follows: Every period the learner go with the majority of the experts that are still alive. If the learner is wrong, he punishes the majority that gave him the wrong by eliminating them. Eventually the the truther expert i^* will be revealed and the learner will always be right. We aim to minimize the number of times the learner is wrong. [4]

[4]Under this type of algorithm the adversary would probably try to make the rate of convergence to the true value as slow as

Majority (Halving) Algorithm

initialize: $V_1 = S$
for $t = 1, 2, \ldots$
receive $\mathbf{x}_t \in \mathcal{X}_t$
predict $p_t = \arg\max_{r \in \{0,1\}} |s_i \in V_t : s_i = r|$
(for tie's predict $p_t = 1$)
receive answer $y_t = s_{i^*}$
update $V_{t+1} = \{i \in V_t : s_i = y_t\}$

Now consider a different formulation of the same algorithm in which we have weights w_i for each expert i. For each i in the majority we set the weight, $w_i = 0$, if the majority is wrong, otherwise we do nothing. Note that in the algorithm below m keeps track of the number of mistakes, and W_m keeps track of the number of experts still alive at mistake m.

Weighted Majority(Halving) Algorithm

initialize: $(\forall i)(w_i \cdot u_i = 1 \cdot u_i)$, $m = 0$, $W_0 = N$
for $t = 1, 2, \ldots$
receive $\mathbf{x}_t \in \mathcal{X}_t$
predict $p_t = \begin{cases} 1 & \text{if } \sum_{\{i:u_i(\mathbf{x}_t)=1 \wedge w_i=1\}} w_i \geq \sum_{\{i:u_i(\mathbf{x}_t)=0 \wedge w_i=1\}} w_i \\ 0 & \text{o.w} \end{cases}$
receive answer $y_t = u_i^*(\mathbf{x}_t)$
update
if $p_t \neq y_t$:
Set $m = m + 1$
$\forall i \in N(u_i(\mathbf{x}_t) \neq y_t \Rightarrow w_i = 0\}$
Set $W_m = \sum w_i$
else $p_t = y_t$:
do nothing

Property of the Majority Algorithm: For a finite set of experts $|U| = N$. The majority algorithm guarantees the mistake bound $M_{\text{majority}} \leq \log_2(N)$.

Proof. Consider the mth mistake then atleast half of the remaining experts were wrong, and so by the updating procedure $W_m \leq \frac{W_{m-1}}{2}$, and so we have that $W_m \leq \frac{W_{m-1}}{2^1} \leq \frac{W_{m-2}}{2^2} \leq \frac{W_{m-2}}{2^3} \leq \cdots \leq \frac{W_{m-m}}{2^m} = \frac{W_0}{2^m} = \frac{N}{2^m}$, thus $W_m \leq \frac{N}{2^m}$. Again we know that there exists a truthful expert u_i^* and so $\forall m(W_m \geq 1)$, it follows that $1 \leq W_m \leq \frac{N}{2^m} \iff 2^m \geq N \iff \log_2(N) \geq m$. Since this holds for all m this holds for maximum, and so $M_{\text{majority}} \leq \log_2(N)$. □

1.3 Classification, Randomized Prediction and Expected Regret

Realizability was a strong assumption so let's not make it. Our goal without any assumptions in this framework is to minimize regret relative to the best expert: $max_{u \in U}\text{Regret}_T(u)$ (*aka min max regret*).
Maximum Regret:

$$\text{Regret}_T(U) = \sum_{t=1}^{T} l(p_t, y_t) - \min_{u \in U} \sum_{t=1}^{T} l(u(x_t), y_t) = max_{u \in U}\text{Regret}_T(u) \tag{5}$$

If there are no assumptions an adversarial environment can make the number of mistakes of any online algorithm equal to T.

possible. How?

Covers Impossibility Result

Without restrictions on the environment an adversarial environment can make a sublinear regret bound impossible.[a]

Sublinear regret bound :

$$\frac{1}{T}\text{Regret}_T(U) = \frac{1}{T}[\sum_{t=1}^{T} l(p_t, y_t) - \min_{u \in U} \sum_{t=1}^{T} l(u(x_t), y_t)] \to 0, \text{ as } T \to \infty, \tag{6}$$

equivalently

$$max_{u \in U}\text{Regret}_T(u) = o(T)$$

[a]For a simple two experts example, $N = 2$, $U = \{u_0, u_1\}$, where u_0 always returns 0, and always returns u_1=1. The adversary can make y_t the opposite of the prediction p_t, ***simply by waiting for the learners prediction and making the correct answer the opposite of*** p_t, thus the adversary can make the number of mistakes of the learner, T. In contrast for any sequence $\{y_i\}_{i=1}^{T}$ of correct answers, let b be the majority of labels, for instance $\{y_i\}_{i=1}^{5} = \{0, 1, 0, 0, 1\}$ then $b = 0$. The the best expert in this $T = 5$ case h_0 makes at most 2 mistakes, and in general T case the best expert makes at most $\frac{T}{2}$ mistakes. Finally note that $(T - \frac{T}{2}) = \frac{T}{2}$ clearly $\frac{T}{2} \neq o(T)$

Note that implicit to get around covers impossibility result we previously assumed the *realizability assumption* (4) . Now we impose another assumption to get around covers impossibility result, before we impose it let us define a couple terms we have already encountered.

Oblivious Adversarial Environment:

The adversarial environment decides on y_t without knowing the the learners choice p_t, while knowing algorithm the learner will us.

An oblivious opponent is defined by a fixed sequence $\{\mathbf{y}_t\}_{t \geq 1}$ of outcomes determined before the game starts. Then, at time t, the learner makes his prediction p_t and the environment reveals the tth outcome $\mathbf{y}_t$. In this model $\mathbf{y}_t$'s are non random fixed values. The adversary knows your algorithm, so he chooses the sequence of answers $\{\mathbf{y}_t\}_{t \geq 1}$ at the beginning with this knowledge to screw you.

Adaptive Adversarial Environment:

The adversary can choose $\mathbf{y}_t$ depending on which expert we followed in the past.

An adaptive opponent is defined by a sequence of functions $g_1, g_2, \ldots$
where $g_t : \{\mathbf{u}_1, \ldots, \mathbf{u}_T\}^{t-1} \to \mathcal{Y}$ and each outcome y_t is given by $y_t = g_t(p_t, \ldots, p_{t-1})$. The adaptive adversary looks at your past choices/behavior up to round $t-1$, with this information chooses y_t to screw you.

[a]

[a]Both environments ***do not*** allow the adversary to simply wait for the learners prediction and making the correct answer the opposite of p_t

The *oblivious environment* assumption is realistic whenever it is reasonable to believe that the actions of the forecaster do not have an effect on future outcomes of the sequence to be predicted

- Weather Forecasting, predicting a sequence of bit, climate change denial.
- The main example when one cannot reasonably assume that the opponent/environment is oblivious is when a player of a game predicts the other players next move and bases action on prediction. Here, the other players future actions may depend on the action of the player in a complicated known,unknown, partially known way. Like the stock market
- Quantum mechanics would also be an interesting example of when obliviousness cannot be assumed in prediction. So suppose you want to estimate the quantum state of a system. Then you have to measure the

wave function which is a mathematical description of the quantum state. But once you observe the wave function it collapses, the environment senses us. Copenhagen interpretation of the measurement problem in quantum mechanics, posits that something in the act of observation results in the collapse of the wave function.

We now consider the expert under the oblivious adversarial environment. In this environment the adversary will pick an algorithm that tries to screw us. The adversary knows our strategy, and the data from previous rounds, but can only surmise what we will do at time t. In this environment the learner is allowed to randomize his predictions.

Expert Problem under randomization

- Decision space: $S = \triangle(N) = \{\mathbf{w} \in \mathbb{R}^N : \mathbf{w} \geq 0 \wedge ||\mathbf{w}||_1 = 1\}$ is the probability simplex.[a]
- Set of competing experts: $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ are the extreme points of the probability simplex. Aka if you follow advice of a single expert, for instance $\mathbf{u}_1 = (1, \underbrace{0, 0, \ldots, 0}_{\text{n times}})$
- outcome space: $\mathcal{Y} = \{0, 1\}^N$.

  There are $t = 1, 2, \ldots, T$ periods, where at period t the learner selects via an algorithm a distribution $\mathbf{w}_t \in S$. Based on $\mathbf{w}_t$ the learner picks an expert at random according to $\mathcal{P}(p_t = i) = w_i$ (*probability learner picks expert i is w_i*). Then the learner receives the true answer, $\mathbf{y}_t \in \mathcal{Y}$, where $y_t(i)$ is the cost of following the advice of the ith expert (*in this context cost is 0 (correct) or 1 (incorrect)*). Then the learner loss (expected loss) is revealed:

$$l(\mathbf{y}_t, \mathbf{p}_t) = E_{\mathbf{p}_t}(\mathbf{y}_t) = \sum_{i=1}^{N} \mathcal{P}(p_t = i) y_t(i) = \langle \mathbf{w}_t, \mathbf{y}_t \rangle,$$

  and updates the probability vector for the next period based on regret $\mathbf{w}_t$.
- Regret (slightly reformulated)[b]:

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{y}_t \rangle - \min_{u \in U} \sum_{t=1}^{T} \langle \mathbf{u}_t, \mathbf{y}_t \rangle$$

  This implies that the regret of the learner is measured relative to the performance of the strategies which always predict according to the same expert. This follows because the max/min of linear functions are the extreme points, and extreme points correspond 1 expert. [c]

The goal of the Expert Problem under randomization is to minimize regret

[a] The l_p norm is as follows $||\mathbf{w}||_p = (\sum_i |w_i|^p)^{1/p}$

[b] $\langle \mathbf{x}, \mathbf{y} \rangle$ is the dot product $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{N} x_i w_i$

[c] In decision theory a randomized strategy is defined as a probability distribution on the action space. Since we can only pick experts, the action space is the set of experts. If the action space has N pure strategies (a pure strategy is a strategy that sticks with one action forever, in our case we have N experts, and so N pure strategies), a randomized strategy is a vector of probabilities, $\mathbf{p} = (p_1, \ldots, p_N)$. To make a decision we choose a probability distribution from the action space, and take the specific action that results upon flipping a coin of chance. So in the formulation above every period we pick our expert according to an experiment of chance according to $\mathbf{w}_t$ since we don't even know what's going to happen our loss is measured by the expectation. Since we don't know, the adversary doesn't know, and hence we get around cover impossibility result. Also get around this by not letting the adversary observe what the learner choose at time t,and simply picking opposite.

1.3.1 Analyzing Regret under oblivious adversaty

Having settled on the notion of regret, we can ask the following questions.

- How does one design an algorithm which is sub linear? (Called Hannan Consistency)
- What kind of feedback can one use for minimizing regret
  - Recall that with the majority algortihm and the consistent algorithm. We used feedback of the loss function l_t to eliminate incorrect experts.
- What are the best rates of convergence, in terms of T, for the sets S, $\mathcal{Y}$ and different type of feedback l_t
- What kind of feedback can one use for minimizing regret?
- How does the dimension n enter into the bounds?
- What are the minimax optimal algorithms and lower bounds on Regret

Cool questions bro but we aren't going to be able to answer all of these in this section. In this section we will fix T and find an algorithm that guarantee an expected regret of $o(T)$ against oblivious adversary.

To recall our problem:

- T time periods
- $\mathcal{Y} = \{0, 1\}^N$ binary outcomes tell us if expert right or wrong
- N experts, so the set of experts $U = \{u_1(\mathbf{x}_t), \ldots, u_N(\mathbf{x}_t)\}$, $u_i : \mathcal{X} \to \mathcal{Y}$ a function from input space to outcome space *(sometimes $u_i(\mathbf{x}_t)$ referred to as $u_t(i)$).*
- Our loss function is the expected loss, and we measure performance of our algorithm by comparing our expected loss with that of the best expert.

The algorithm that we are going to use to guarantee $o(T)$ regret is randomized weighted majority. Note that because we do not assume *realizability* we can't just eliminate experts that are wrong, because we don't have a guarantee that there exists an expert that will never make a mistake. So instead of eliminating experts that are wrong, we decrease weightings of wrong experts. So the intuition of the weighted majority algorithm is 'if we are wrong then many experts suffer,' which is a lot like the halving algorithm. Before we move on to the next section let us first recall the definition of regret relative to a specific expert.

Regret for expert i is as follows:

$$R_{t-1}(\mathbf{u}_i) = \underbrace{\sum_{j=1}^{t-1} \langle \mathbf{w}_j, \mathbf{y}_j \rangle}_{\hat{L}_{i,t-1}} - \underbrace{\sum_{j=1}^{t-1} \langle \mathbf{u}_i(\mathbf{x}_j), \mathbf{y}_t \rangle}_{L_{i,t-1}}$$

$\mathbf{u}_i(\mathbf{x}_j)$ is the ith expert choice at period j

$L_{i,t-1}$ cumulative loss of expert i at period $t-1$

$\hat{L}_{t-1}$ expected cumulative loss of expert learner at period $t-1$

1.3.2 Weighted Average Algorithms

Weighted Average Algorithm

parameter: $\eta \in (0,1)$
initialize: $(\forall i)(\theta_0(i) = 0)$
for $1, 2, \ldots$
choose based on $\mathcal{P}(p_t = i) = w_t(i)$, where $w_t(i)$ is the prob ith expert is chosen
receive cost of all experts $\mathbf{y}_t \in \mathcal{Y} = \{0,1\}^N$
update rule compute weightings and probabilities for the next period
compute weighting $\theta_{1,t}(\eta)$, compute probability vector $\mathbf{w}_{t+1}$, so $\forall i, w_{t+1}(i) = \frac{\theta_{i,t}(\eta)}{\sum_{j=1}^N \theta_{j,t}(\eta)}$

The weights for round $\theta_{1,t}(\eta), \ldots, \theta_{N,t}(\eta) \geq 0$ are assigned[a] to each expert at the end of period t, and the weighting of expert i is based on the regret of not following expert i in the previous period, $t-1$.If regret in period $t-1$ is large, $R_{t-1}(\mathbf{u}_i)$, then we assign large weight $\theta_{i,t}(\eta)$ to expert i the next period.

$$R_{t-1}(\mathbf{u}_i) \uparrow \Rightarrow \theta_{i,t}(\eta) \uparrow$$

[a] I might have gotten the indexes messed up. Compute the weights for the next round, at the end of the round before. These weights are based on the regret realized at the end of the round before

The general rule under this algorithm is to give more weight to experts when the cumulative regret from not following them in the previous period is large. Thus we can think of the weight $\theta_{i,t-1}(\eta)$ as an arbitrary strictly increasing function of the expert's regret. It's convenient to write this function as the derivative of a nonnegative, convex, and increasing function(Aka strictly convex and positive [5]) $\phi : \mathbb{R} \to \mathbb{R}$, where the learner uses ϕ' to determine the weight $\theta_{i,t}(\eta) = \phi'(R_{t-1}(\mathbf{u}_i), \eta)$ assigned to the ith expert. Denote $R_{i,t-1} := R_{t-1}(\mathbf{u}_i)$.

Weighted Average Learner The learner predicts at time t according to the normalized weight

$$w_t(i) = \frac{\phi'(R_{i,t-1}, \eta)}{\sum_{j=1}^N \phi'(R_{j,t-1}, \eta)} \tag{7}$$

where weights $\phi'(R_{i,t-1}, \eta)$ used at time t assigned to each expert at the end of round $t-1$, aka the learner choose weights according to regret up to time $t-1$.For weighted majority algorithm's usually all we need to do is find positive, increasing, convex functions of the learners regret to represent the weightings.
Instantaneous Regret The instantaneous regret with respect to expert i at time t:

$$r_{i,t} = l(\mathbf{w}_t, \mathbf{y}_t) - l_t(i), \text{ Where } l_t(i) = l(\mathbf{u}_i(t), \mathbf{y}_t) \text{ is the loss of expert } i \text{ at the end of round } t$$

This is the regret the learner feels of not having listened to expert i right after the tth outcome[a] $\mathbf{y}_t$ is revealed. Note that $R_{i,t} = \sum_{j=1}^t r_{i,j}$.

[a] whoops I meant 'right after the tth loss is revealed'

[5] the function is positive then the derivatives are positive

In the next example $\phi'(R_{i,t-1},\eta) = e^{\eta R_{i,t-1}}$, and so $w_t(i) = \frac{e^{\eta R_{i,t-1}}}{\sum e^{\eta R_{i,t-1}}}$ we can verify [6]that $w_t(i) = \frac{w_{t-1}(i)\cdot e^{-\eta l_t(i)}}{\sum_{j=1}^N w_{t-1}(j)\cdot e^{-\eta l_t(j)}}$

Randomized Exponential Weighted Average

Abuse of notation: The weights for each expert at the end of period t will be $w_t(i) = w_{t-1}(i)\cdot e^{-\eta l_t(i)}$ and the probability that the learner will pick experts i for answer p_t will be $\mathbf{w}_t(i) = \frac{w_{i,t-1}e^{-\eta l_t(i)}}{\sum_{j=1}^N w_{j,t-1}e^{-\eta l_t(j)}}$, where $\mathbf{w}_t$ is the probability vector.

For each i we set $w_t(i) = w_{t-1}e^{-\eta l_t(i)}$.
initialize: $(\forall i)(w_0(i) = 1)$
for $t = 1, 2, \ldots$
receive $\mathbf{x}_t \in \mathcal{X}_t$
predict: randomly according to probability vector $\mathbf{w}_t$,
where $\mathbf{w}_t(i) = \frac{w_{i,t-1}e^{-\eta l_t(i)}}{\sum_{j=1}^N w_{j,t-1}e^{-\eta l_t(j)}} = \mathcal{P}(p_t = i)$
receive: $\mathbf{y}_t \in \mathcal{Y}$, where $y_t(i)$ loss of following ith expert
update: the weights of experts, $w_t(i) = w_{t-1}e^{-\eta l_t(i)}$, and probability vector,$\mathbf{w}_t$.

So if expert i is wrong we reduce their weight, and as the parameter η increases the punishment(reward) for being wrong(right) increases as well, visa-versa.

Before we prove the bound for randomized exponential weighted algorithm theorem let's prove a useful lemma that will be used in it.

Lemma: For η and x small enough $log(E[e^{-\eta x}]) \leq 1 - \eta E[x] + \frac{\eta^2}{2}E[x^2]$

Proof. By the taylor expansion

$$e^{-\eta x} = 1 - \eta x + \frac{\eta^2}{2}x^2 + \ldots$$

Thus when η and x are sufficiently small we have:

$$e^{-\eta x} \leq 1 - \eta x + \frac{\eta^2}{2}x^2$$
$$E[e^{-\eta x}] \leq 1 - \eta E[x] + \frac{\eta^2}{2}E[x^2]$$

Using the fact that[7] $log(1+x) \leq x$ we have:

$$log(E[e^{-\eta x}]) \leq 1 - \eta E[x] + \frac{\eta^2}{2}E[x^2]$$

[6]

$$w_t(i) = \frac{e^{\eta R_{i,t-1}}}{\sum_{j=1}^N e^{\eta R_{j,t-1}}} = \frac{e^{\eta(\hat{L}_{t-1}-L_{i,t-1})}}{\sum_{j=1}^N e^{\eta(\hat{L}_{t-1}-L_{j,t-1})}} = \frac{e^{\eta\hat{L}}\cdot e^{-\eta L_{i,t-1}}}{e^{\eta\hat{L}}\sum_{j=1}^N e^{-\eta L_{i,t-1}}} = \frac{e^{-\eta L_{i,t-1}}}{\sum_{j=1}^N e^{-\eta L_{i,t-1}}} = \frac{e^{-\eta\sum_{k=1}^{t-1} l_k(i)}}{\sum_{j=1}^N e^{-\eta\sum_{k=1}^{t-1} l_k(i)}},\ \text{and}$$

$$w_{i,t+1} = \frac{e^{-\eta\sum_{k=1}^{t} l_k(i)}}{\sum_{j=1}^N e^{-\eta\sum_{k=1}^{t} l_k(i)}} = \frac{e^{-\eta\sum_{k=1}^{t-1} l_k(i)}\cdot e^{-\eta l_t(i)}}{\sum_{j=1}^N e^{-\eta\sum_{k=1}^{t-1} l_k(j)}\cdot e^{-\eta l_t(j)}} = \frac{(e^{-\eta\sum_{k=1}^{t-1} l_k(i)}\cdot e^{-\eta l_t(i)})/\sum_{j=1}^N e^{-\eta\sum_{k=1}^{t-1} l_k(j)}}{(\sum_{j=1}^N e^{-\eta\sum_{k=1}^{t-1} l_k(j)}\cdot e^{-\eta l_t(j)})/\sum_{j=1}^N e^{-\eta\sum_{k=1}^{t-1} l_k(j)}}$$
$$= \frac{(e^{-\eta L_{i,t-1}}\cdot e^{-\eta l_t(i)})/\sum_{j=1}^N e^{-\eta L_{j,t-1}}}{(\sum_{j=1}^N e^{-\eta\sum_{k=1}^{t-1} l_k(j)}\cdot e^{-\eta l_t(j)})/\sum_{j=1}^N e^{-\eta L_{j,t-1}}} = \frac{w_{i,t}\cdot e^{-\eta l_t(i)}}{\sum_{j=1}^N w_{j,t}\cdot e^{\eta l_t(j)}}$$

[7]Two important inequalities to remember: $1 - x \leq e^{-x}$ and $1 + x \leq e^x$. Taking the log yields the result $log(x+1) \leq x$. The inequality $1 + x \leq e^x$ is in words means that the line $y = 1 + x$ is tangent to the curve $y = e^x$. George Polya actually thought of this

□

Theorem: Bound for on Regret for Randomized Exponential Weighted Algorithm

$$\hat{L}_T - \min_i \sum_{t=1}^{T} l_T(i) \leq \frac{log(N)}{\eta} + T\eta \tag{8}$$

Note that we can make η as small as we want. So if we want sublinear regret, we should require that $\eta^{-1}\ln(N) \to 0$ as $T \to \infty$. Existence of such a condition is Hannan's Consistency Theorem (4.1 in Lugosi book)
Most important part: If we choose $\eta = \frac{\log N}{T}$, then we have

$$\hat{L}_T - \min_i \sum_{t=1}^{T} l_T(i) \leq \sqrt{T \log N}$$

- This proof has the same flavor as the weighted majority algorithm, in that algorithm every expert started w/ weight one, and if the learner made a mistake then reduce the weight of the majority to 0. So the argument in that case was 'either we are correct or many experts suffer.' So we look at the sum of all the weights every period, and argue that either we are correct in period t or the sum of all the weights at the end of period t, W_t, goes down by alot

Proof. Let

$$W_t = w_t(1) + w_t(2) + \cdots + w_t(N),$$

be the sum of the weights at the end of period t. Since $w_t(i) = w_{t-1}(i)e^{-\eta l_t(i)}$ we can substitute this in for $w_t(i)$ yielding

$$W_t = \underbrace{w_{t-1}(1)}_{\text{weight}} \cdot \underbrace{e^{-\eta l_t(1)}}_{\text{punishment}} + w_{t-1}(2) \cdot e^{-\eta l_t(2)} + \ldots w_{t-1}(N) \cdot e^{-\eta l_t(N)}.$$

Now note that probability at period t that expert i is chosen is $p_t(i) = \frac{w_{t-1}}{\sum_{k=1}^{N} w_{t-1}(k)}$, and so $p_t(i) \cdot \sum_{k=1}^{N} w_{t-1}(k) = w_{t-1}(i)$, substituting this in for $w_{t-1}(i)$ we have

$$\begin{aligned}
W_t &= p_t(1) \cdot \sum_{k=1}^{N} w_{t-1}(k) \cdot e^{-\eta l_t(1)} + \cdots + p_t(N) \cdot \sum_{k=1}^{N} w_{t-1}(k) \cdot e^{-\eta l_t(N)} \\
&= \sum_{k=1}^{N} w_{t-1}(k)[p_t(1) \cdot e^{-\eta l_t(1)} + \cdots + p_t(N) \cdot e^{-\eta l_t(N)}] \\
&= W_{t-1}[\underbrace{p_t(1) \cdot e^{-\eta l_t(1)} + \cdots + p_t(N) \cdot e^{-\eta l_t(N)}}_{(**)}] \\
&= W_{t-1} \cdot E[e^{-\eta X_t}] \text{ (where the rando variable } X_t = \textit{learner's loss at time } t)
\end{aligned}$$

The underlined part is the expected value $e^{-\eta(\text{my loss})}$ at period t (aka $E_t[e^{-\eta \text{loss}}]$) this follows since the expected loss of the learner at period t is $E[loss]_t = \sum_{k=1}^{N} p_t(k) \cdot l_t(k)$.

Now if the learner's loss is large, since e^{-x} is decreasing in it's argument, we have that (**) will be small, aka the experts weights are reduced by alot. So every time that the learner's loss is large, the total weight (total

relationship in a dream and used it to prove the AM-GM inequality. When asked about his proof years later Polya claimed it was *the best mathematics he had ever dreamt of* (See pg. 23 in 'The Cauchy-Schwartz Master Class'). Note this also follows by another very important relation $(\forall r > 0)[log(x) \leq x^r]$ (*the log of x is less than any positive power of x*) since $E[e^{-\eta x}] \leq 1 - \eta E[x] + \frac{\eta^2}{2}E[x^2]$ we have $log(E[e^{-\eta x}] \leq E[e^{-\eta x}] \leq 1 - \eta E[x] + \frac{\eta^2}{2}E[x^2]$.

reputation) of the expert goes down by alot.

We now want to relate the weight of experts to their loss. So let's fix a rando expert i recall

$$w_T(i) = w_{T-1}e^{\eta l_T(i)} = w_{T-2}e^{\eta l_{T-1}(i)}e^{\eta l_T(i)} = \cdots = \prod_{k=1}^{T} e^{-\eta l_k(i)} = e^{-\eta[l_1(i)+\cdots+l_T(i)]} = e^{-\eta L_T(i)}.$$

Now note that for each period t the weight of the expert is less than the total weight of all the experts in period t, and so we have the following:

$$\begin{aligned} w_1(i) = e^{-\eta l_1(i)} &\leq W_1 = W_0 \cdot E[e^{-\eta X_1}] = N \cdot E[e^{-\eta X_1}] \text{ (since total weight at time 0 is N)} \\ w_2(i) = e^{-\eta l_1(i)+l_2(i)} = e^{-\eta L_2(i)} &\leq W_2 = W_1 \cdot E[e^{-\eta X_2}] = N \cdot E[e^{-\eta X_2}] \cdot E[e^{-\eta X_1}] \\ &\vdots \\ w_T(i) = e^{-\eta L_T(i)} &\leq N \cdot \prod_{k=1}^{T} E[e^{-\eta X_k}]. \end{aligned}$$

Now let's take the log of this equation

$$-\eta L_T(i) \leq log(N) + \sum_{k=1}^{T} log(E[e^{-\eta X_k}]).$$

We want switch the log with the expectation, since the would simply be the learners loss. Can't do this but we can do something similar with the lemma we just proved Equation 1.3.2.

From the lemma if η is small enough we have:

$$\begin{aligned} -\eta L_T(i) \leq log(N) + \sum_{k=1}^{T} log(E[e^{-\eta X_k}) &\leq log(N) - \eta \sum_{k=1}^{T} E[X_k] + \frac{\eta^2}{2} \sum_{k=1}^{T} E[X_k]^2 \\ &\leq log(N) - \eta \sum_{k=1}^{T} E[X_k] + \frac{\eta^2}{2} \sum_{k=1}^{T} 1^2 \text{ (since loss } x_t \text{ is } 0 \leq x_t \leq 1) \\ &= log(N) - \eta \sum_{k=1}^{T} E[X_k] + \frac{\eta^2}{2} T \\ &= log(N) - \eta \hat{L}_T + \frac{\eta^2}{2} T \\ \eta(\hat{L}_T - L_T(i)) &= \frac{\eta^2}{2} T + \log(N) \\ \hat{L}_T - L_T(i) &= \frac{\eta}{2} T + \frac{\log(N)}{\eta} \end{aligned}$$

As required.

□

We now show another property concerning the asymptotic tightness of bounds of a randomized strategy in an adversarial environment, but before we proof and define this property let's prove a useful lemma that will be used in the proof.

Lemma Let $\{X_i\}_{i\geq 1}$ be a sequence of independent identically distributed [a] [b] then

$$E[\min\{X_i\}] < \min_i E[X_i]$$

[a] For r.v.s X and Y, $X \leq Y$ in the usual stochastic order if $(\forall x \in \mathbb{R})[P(X > x) \leq P(Y > x)]$, $X \leq_{st} Y$

[b] Strict inequality doesn't hold for every independent r.v.s but it does hold when their supports are note disjoint (in particular, when they are identically distributed)

Proof. Simple case X_1 and X_2 i.i.d r.v.s. Let $U = \min\{X_1, X_2\}$ we want to show that

$$EU < EX_1 \wedge EU < EX_2$$

then we can conclude that $EY < \min\{EX_1, EX_2\}$.
We have

$$\begin{aligned} P(U > b) &= P(X_1 > b, X_2 > b) \\ &= P(X_1 > b)P(X_2 > b) \text{ (by independence)} \\ &= P(X_1 > b)^2 \text{ (by identical distributed)} \\ &< P(X_1 > b) \text{ (since } \forall b[0 < P(X_1 > b) < 1]) \text{ ,} \end{aligned}$$

and so

$$\int_0^\infty P(U > b) < \int_0^\infty P(X_1 > b) \iff E[U]^+ < E[X_1]^+.$$

If X_1 was positive then we would be done, so suppose not. By looking at $-P(U \leq b)$ and $-P(X_1 \leq b)$ we have:

$$[1 - P(U > b)] = [1 - P(X_1 > b)^2] > [1 - P(X_1 > b)] \text{(by iid, and since } \forall b[0 < P(X_1 > b) < 1])$$
$$\Rightarrow \int_{-\infty}^0 P(U \leq x)dx > \int_{-\infty}^0 P(X_1 \leq x)dx \iff E[U]^- > E[X_1]^- \iff -E[U]^- < -E[X_1]^-$$

and so $E[U] = E[U]^+ - E[U]^- < E[X_1]^+ - E[X_1]^- = E[X_1]$ if we do the same thing for X_2 the result follows, and so $E[U] < \min\{EX_1, EX_2\}$.[8] □

Corollary 4.1?: Asymptotic Tightness of bounds with N experts:

An oblivious adversary can always cause us regret that is atleast $\sqrt{\log(N) \cdot T}$

$$\hat{L}_T - \min_i L_T(i) \geq b \cdot \sqrt{\log(N) \cdot T},$$

where b is just some fixed constant
In particular with 2 experts an oblivious adversary can always cause us regret that is atleast $b\sqrt{T}$

$$\hat{L}_T - \min_i L_T(i) \geq b \cdot \sqrt{T}$$

We prove the 2 expert case below, and leave the general case as an exercise. To prove the general note that the expectation of the min of N std normal random variables is bounded by something that has to do with $\sqrt{\log(N)}$. Please refer to `http://www.gautamkamath.com/writings/gaussian_max.pdf` for the proof.

Note that if we can get $b\sqrt{\log(N) \cdot T} \to 0$ as $T \to 0$ then we would have hannan consistency.

Proof. Suppose we have 2 experts, $N = 2$,and the adversary makes each expert fail at the flip of a coin for each round t. The expected loss of the learner is $T/2$, and distribution of expert i is $bernoulli(1/2)$, let X_i represent the random variable of the learner's loss. Now consider $Y_i = \sum_{i=1}^T X_i$ were Y_i is the total loss of expert i up until T. Since X_i is Bernoulli, Y_i $binomial(T, 1/2)$ and so the expected cumulative loss of expert i

[8] For random variable X, $E[X] = \int_0^\infty (1 - F(x))dx - \int_{-\infty}^0 F(x)dx$. Wiki 'expected value' if unfamilier

at round T is simply $T/2$, the variance [9][10] of Y_i is $T/4$. By the Lindberg-Levy Clt we have that at the limit (*when the number of periods T is very large*) $\frac{Y_i - T/2}{\sqrt{1/4}} \to^d \mathcal{N}(0,1)$, and so $\sqrt{T/4} \cdot \frac{Y_i - T/2}{\sqrt{T/4}} \sim \mathcal{N}(0, T/4)$ and finally $Y_i = T/2 + \sqrt{T/4} \cdot \frac{Y_i - T/2}{\sqrt{T/4}} \sim \mathcal{N}(T/2, T/4)$. Denote Z_i to be the std normal rando variable then $Y_i = T/2 + \sqrt{T/4} \cdot Z_i$. Therefore

$$E[\min\{Y_1, Y_2\}] = T/2 + \frac{\sqrt{T}}{2} \underbrace{E[\min\{Z_1, Z_2\}]}_{b},$$

but recall $\min_i E\{Z_1, Z_2\} < E[\min\{Z_1, Z_2\}] = 0$ *(since $\{Z_1, Z_2\}$ are iid mean zero std normal rvs)*, therefore $b < 0$ so the term on the RHS is negative. Recall that the expectation of the learners loss is $\frac{T}{2}$, and so the expectation of the best expert, $E[\min\{Y_1, Y_2\}]$, is far away from $\frac{T}{2}$ by some constant, $b/2$, times $\sqrt{T}$.

Rearranging the equation, while taking note that the RHS is negative we can conclude that when the *adversary is not even trying*, just flipping a coin to screw with both the experts and the learner we have that:

$$\hat{L}_T - \min_{i \in N} L_T(i) = T/2 - E[\min\{Y_1, Y_2\}] = \sqrt{T}\frac{b}{2}.$$

Therefore we have our result since with the teeniest bit of effort the adversary can make us have regret $\hat{L}_T - \min_{i \in N} L_T(i) \geq \sqrt{T}\frac{b}{2}$

Let's combine the theorem and the corollary we just proved (we just proved 2-expert case, but lets assume we did the general case) , then we have that for η small enough

$$b \cdot \sqrt{\log(N) \cdot T} \leq \hat{L}_T - \min_i L_T(i) \leq \frac{log(N)}{\eta} + T\eta$$

Using mathetematica $b \cdot \sqrt{\log(N) \cdot T} = \frac{log(N)}{\eta} + T\eta$ whenever

$$\eta = \pm \frac{\sqrt{b^2 - 4}\sqrt{T}\sqrt{\log N} - b\sqrt{T \log N}}{2T} = \frac{\sqrt{b^2 - 4}\sqrt{\log N}}{2\sqrt{T}} - \frac{b\sqrt{\log N}}{2\sqrt{T}}.$$

Let's restrict η to be positive then if $b \geq 0$ we can set $\eta = \frac{(\sqrt{b^2-4} - b)\sqrt{\log N}}{2\sqrt{T}}$. But η needs to be small for this to hold, but if we focus on the positive solution, for T large enough $\eta \to 0$ and so we have proved $o(T)$ regret. [a]

[a] Recall from 8 that if we choose $\eta = \frac{\log N}{T}$, then the upper is $\sqrt{T \log N}$, there for when $\eta = \frac{\log N}{T}$ we have that ...

□

1.3.3 Let me wrap this section up with a monologue

To recap an oblivious adversary knows the learner's algorithm, he can pick the sequence of answers in advance, however he doesn't use the past realizations of our experts answers, and he can't view the learner's answer at round t then change y_t so that the learner is wrong (the adversarial adversary is restricted from doing this as well). When the oblivious adversary decides which expert to give losses to at time t he doesn't know which expert we decided to follow at time $t-1$. He does know how we randomized, but he doesn't know which expert we

[9] For the general version

[10] Note for iid bernoulli the variance is $var(X) = EX^2 - [EX]^2 = 1^2 \cdot (1/2) + 0^2 \cdot (1/2) - 1/4 = 1/2 - 1/4 = 1/4$, thus the variance of Y_i is $T/4$

actually picked at round $t-1$.

Now let's consider the adaptive adversary, this guy can respond to what we do Equation 1.3. Under a randomized strategy would it be better for the adaptive adversary if he could respond to what the learner did in the past? No for two reasons:

1. In the randomized weighted average algorithm. The learner didn't use his own past choices. The learner's choices every day depended only on what the expert does, and didn't depend on what the learner did.
    - So in our algorithm p_t depended on the losses the adversary gave us in the past, through our updating procedure on the weights of the experts. But it p_t didn't depend on which expert we chose in the past (since the choice was completely random).
2. Now let's put ourselves in the shoes of the adaptive adversary, we know the learners strategy, and we know that the learner plays a strategy that doesn't look back at past choices. Since the learner isn't looking back the we only cares about how well the learner randomizes in the next period. How well the learner randomizes in the next period depends, on what we did in the previous period. So the way we screw the learner is by looking forward and not back.
3. Now let's put ourselves in the shoes of the oblivious adversary. We know that the distribution of how the learner plays is a response to what we did in the past. So we can imagine the worst distribution, output a sequence of answers s.t the learner has the worst distribution. We can do this because we know the players updating algorithm. Then for every period the distribution of the learners true action is the same distribution that the we 'imagined' the learner to play.
4. Thus we can see that under randomization the oblivious and adaptive adversary act the same
    - The intuition is similar to the bellman equation. The oblivious adversary picks a sequence at time 0 s.t the leaner plays 'shittly' (minimizing payoff of learner) for time infinity and beyond. The adaptive adversary minimizes payoff every period (bellman). (Care's about state he is in now, doesn't care about the past) Both are the same. The loss function is additive, and so this argument actually works, since bellman type arguments work when losses/rewards are additive from period to period.

Exercise 1: Consider a matrix game with two player. Assume each player only knows about his own payoff and not the others. Think of every column of the matrix game as an expert, at each realization of what player i does a reward/loss is revealed. Now play some no regret strategy s.t after a million periods player i doesn't say 'damn I could have played better'[a]. Suppose both players do this, what will happen? At the end there will be some distribution, what can be said about this distribution? Relate this to the original matrix game problem and equilibrium notions.

[a] regret being $o(T)$

2 Online Convex Optimization

Recall the expert problem under randomization section 1.3. We picked $\mathbf{w}_t \in \triangle(N)$ and we had linear loss function $l_t = \langle \mathbf{w}_t, \mathbf{y}_t \rangle$ which was just the expected loss for period t. In online convex optimization we pick $\mathbf{w}_t \in S$ where S is now a general convex compact set, and l is now an arbitrary convex function, $U \subseteq S$ we have this condition because if $S \subset U$ then there could exists $\mathbf{u}$ optimal st $\mathbf{u} \notin S$ then we could never get close to $\mathbf{u}$ every period we would regret more and more. [11] Online convex optimization is very similar to static optimization except that every period we will have to make a choice and every period we will know a bit more of the objective function.

[11] This section is based heavily on Shai Shalev-Shwartz survey of online learning

General Online Convex Optimization

constraint set: S, hypothesis set U
Input A convex *constraint set* S, loss function l, hypothesis class U
for $t = 1, 2, \ldots, T$
predict vector $\mathbf{w}_t$
environment gives loss function $l_t : S \to \mathbb{R}$
suffer loss $l(\mathbf{w}_t)$
compare loss to loss suffered from in some hypothesis class U
Maximum Regret:

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^{T} l(\mathbf{w}_t) - \min_{\mathbf{u} \in U} \sum_{t=1}^{T} l(\mathbf{u})$$

Where $\mathbf{u} \in U$. This is the regret we would get if in retrospect we could have chosen u. The very left term is an offline optimization term. It is the case when we know loss functions l_t in advance and optimize from $t+1$ to T using usual static optimization techniques.
Find algorithm that minimizes maximum possible regret for all u in some set U.

So at the end of period T after we have chosen $\mathbf{w}_T$ and received loss l_T, then the sequence of loss functions $\{l_t\}_{t=1}^{T}$ have been revealed. Since we have these functions we can perform the offline optimization to see what the best action would be given these losses. Usually $S = U$ the set we choose online, S, and the set we choose offline U. Sometimes it's convenient for the sets to differ

2.1 Examples of Online Convex Optimization

This section will go through some examples of online convex optimization. This material is not new we have seen it before. For instance in the expert problem. $S = \triangle(N)$ and $w_t(i) \in S$ was the probability we chose expert i, our loss function $l_t = \langle \mathbf{w}_t, \mathbf{y}_t \rangle$ was the expected value which is linear/convex. Losses where also uniformly bounded since $\mathcal{Y} = \{0, 1\}$, and so $0 < |l_t(i)| \leq 1$. In this problem we showed that for the oblivious adversary weighted exponential algorithm gives regret is bdd by $\Omega(\sqrt{t \log n})$ when the losses are uniformly bounded. [12]

2.1.1 Online Regression

Offline
Suppose want to predict the weight of a child 20 years from now, if we know weight at birth, height at birth, gender, etc.:
N explanatory variables
1 variable to predict
Observations $1 \leq t \leq T$
dataset $X_t(1), \ldots, X_t(n), Y_t$.
So the $X's$ are the weight at birth, height at birth, etc. and Y_t weight at age
We choose scalars $w(1) \ldots w(n)$ to predict y_t as a linear combination of data, $\{X_t\}$.
Our goal is to find values that minimize some convex loss function:

We would make prediction according to:

$$\sum_{t=1}^{T} l(w(1)x_t(1) + \cdots + w(n)x_t(n), y_t).$$

[12] The notation $\Omega(f(t))$ means that for t large enough we have that $k \cdot f(t)$ for some canstant k. This is a computer science term (please refer to `https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-big-omega-notation`).

In regression functions we usually have loss quadratic loss function, $l_t = (\langle \mathbf{x}_t, \mathbf{w} \rangle - \mathbf{y}_t)^2$, this is the FOC/Inverse solution, alternative loss function is absolute loss, $l_t = |(\langle \mathbf{x}_t, \mathbf{w} \rangle - \mathbf{y}_t)|$ this is a linear programming problem.

Online
for $t = 1, \ldots, T$
Learner chooses $\mathbf{w}_t \in S$
Adversary announces $\mathbf{x_t}$, and y_t.
Then the loss is revealed $l_t = (\underbrace{\langle \mathbf{w}_t, \mathbf{x}_t \rangle}_{\text{prediction}}, \underbrace{y_t}_{\text{actual}})$.

We assume loss is convex. For the quadratic loss the reference set U is the regular OLS over all past values. So in this problem we minimize max regret:

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^{T} l_t(\mathbf{w}_t) - \underbrace{\min_{\mathbf{u} \in U} \sum_{t=1}^{T} l_t(\mathbf{u})}_{\text{online problem}}$$

Regret is relative to the *online problem* mentioned in the 'online problem' description above, and underlined in the equation.

2.1.2 Online Spam Filtering

Emails arrive into system classified as spam or valid. The generator's of this spam emails know generally what type of spam filtering algorithm is used and is trying to get around this, thus we are in an *adaptive adversarial environment.*

Each email is represented as a vector $\mathbf{a} \in \mathbb{R}^N$, where N is the number of words in the dictionary. The entries of this vector are all zero, except for coordinates that correspond to words appearing in the email, which are assigned values 1.

To predict spam we learn a filter[13] , $\mathbf{u}$ in the constraint set S, where $S \subset \mathbb{R}^N$. Classification of an email $\mathbf{a} \in \mathbb{R}^N$ by a filter $\mathbf{u} \in S \subset \mathbb{R}^N$ is given by the sign of the inner product between these two vectors, $p_t = sign\langle \mathbf{x}, \mathbf{a} \rangle$ so $+1$ means valid, -1 means spam.[14]

The hypothesis set $U = S$ is space of norm-bounded linear filters[15]. Losses are determined by stream of incoming emails, and the labels (valid or spam) could be known, partially known, unknown. Let $(\mathbf{a}, y)$ be a email/label pair. Then the cost for an arbitrary filter(expert) $\mathbf{u}$ is $f(\mathbf{u}) = l(\hat{y}, y)$ where $\hat{y}$ is the classification given given by filter, and y is the true label, and l is a convex loss function (like the square loss $l(\hat{y}, y) = (\hat{y} - y)^2$).

This is similar expert problem. The space of filters, U, are the experts. We predict according to them, and minimize regret every period according to some convex loss function.

2.1.3 Netflix/Spotify Recommendation systems: matrix completion model

The customers are rows, the different media are columns, and each entry corresponds to a particular user/media pair. The value in each entry is a value scoring the preference of the user for that particular media.

For instance for binary recommendations like music $X \in \{0, 1\}^{n \times m}$, n number of people considered, m number of songs in our library, 0-1 means like-dislike:

$$X_{i,j} = \begin{cases} 0 & \textit{hate song} \\ 1 & \text{o.w} \end{cases}$$

[13] learn based on all the users in our system, who indicate the email is spam, delete the email, don't even open the email. Once the does this we get data. Or we could learn by putting what we think is spam in the spam folder, then the user reveals to us the true answer by placing it in the correct folder

[14] So $\mathbf{u}$ is a vector of -1 and 1 if the email matches w/ too many spammy words, and spammy words out weigh the non-spammy, then the sign of the dot product will be negative

[15] So a filter is a ball, if the distance between the email and any point in the ball is less than or equal to the radius of the ball then classify the email as spam!

At each iteration the learner picks a preference matrix from the set of preference matrices $X_t \in S \subset \{0,1\}^{n\times m}$, (constraint set) which is subset of all possible $0-1$ matrices.

Each iteration t we choose a constraint set $P_t \in S$, and then the adversary chooses a user-song pair (i_t, j_t) along w/ a 'real' [16] pair $y_t \in \{0,1\}$. The loss by the decision maker is $l_t(P_t) = (P_{i_t,j_t} - y_t)^2$. The hypothesis set, U, is the set of all low rank matrices, so we calculate regret relative to low rank matrix meaning performing, on average, as few preference-prediction errors as the best low-rank matrix. [17].

Compare this to the linear model, before we restricted ourself to predicting y_t as a linear combination of the data, and based on theory we thought it to be valid to restrict ourselves to this set of linear predictors. In this model for practical reasons we restrict ourselves to a low-rank matrix and try to explain users choices by a few key factors by making the hypothesis class U the set of low-rank matrices.

2.1.4 Universal Portfolio

N assets, $U = S = \triangle(N)$, interpret these sets as the pool of investing strategies.
for every $t = 1, \ldots, T$

Choose: $\mathbf{w}_t \in S$, $w_t(i)$ is the fraction of money invested in asset i

adversary announces: return $\mathbf{q}_t$

$\mathbf{q}_t$ *has strictly positive entries s.t* $q_t(i)$ *is the price ratio for the* i*th asset between iterations* t *and* $t+1$

receive reward: $r_t(\mathbf{w}_t, \mathbf{q}_t) = \log(\langle \mathbf{w}_t, \mathbf{q}_t \rangle)$

This follows since the ratio between wealth at $t+1$ *and* t *is* $\langle \mathbf{w}_t, \mathbf{q}_t \rangle$ *and so the reward is the log of this change ratio in wealth* $\log(\langle \mathbf{w}_t, \mathbf{q}_t \rangle)$

We again aim to minimize regret:

$$\text{Regret}_T(\mathbf{u}) = \underbrace{\max_{\mathbf{u}\in U} \sum_{t=1}^{T} r_t(\mathbf{u}, \mathbf{q}_t)}_{\text{online problem}} - \sum_{t=1}^{T} r_t(\mathbf{w}_t, \mathbf{q}_t)$$

Note that in the online problem the first entry is fixed, so we are comparing our return relative to a strategy that keeps the portfolio's balance constant over time, and hence, this term is called a constant rebalanced portfolio. The second term is wealth accumulated by the learner that is rebalanced period to period. *Universal* portfolio means that regret is $o(T)$.

2.1.5 Wrap up of examples

- So in these types of classification problems the inputs are very sparse. For the email we have a vector the length of the diction of 1 and 0 for the appearance of a word in a specific email. In the netflix example, we have a vector of length entire movie/show database, very sparse. The important thing is that in these examples we make predictions and learn at same time. Contrary to offline problem, in which we have a data set/training set an make prediction on another set[18].

- If we want to maximize some reward instead of minimizing cost the regret is slightly reformulated, as in the portfolio example. Instead of convex loss function we need a concave reward function.

- The hypothesis set U differs from case to case. It can be based on theory, so if the offline program is a LP then U are the extreme points. It can be based on practicality, in the recommendation example describing users by a few factors is computationally easier than the alternative.

[16] the actual scoring scoring for the song user pair

[17] low rank means everything is a linear combination of a couple basis vectors, meaning each entry in the matrix. Only a few factors that determine a consumer preferences for music. determined by a few unknown factors. We basically trying to describe the consumer by a couple indicators

[18] Manski: 'prediction off the support'

2.2 Gradient Descent

First we'll take a quick look at the T-series expansion. Then actually define convexity and get a few properties using the T-series expansion that we will need for the gradient descent algorithm which we will look at last.

2.2.1 T-Expansion

Mean Value Theorem: f is continuous on $[a,b]$ (closed) and differentiable on open interval (a,b), then exists a point (a,b) such that: $f^{'}(c) = \frac{f(b)-f(a)}{b-a}$

Taylor Expansion[19] Expanding out the mean value thm, we have $f(b) = f(a) + (b-a)f'(c)$, let $h = b-a$, then c lies between b and a so it must be the case that $\exists t \in (0,1)$ s.t

$$c = \underbrace{ht}_{\text{multiple of interval}} + \underbrace{a}_{\text{starting point}},$$

and so we have

$$f(a+h) = f(a) + hf'(a+th).$$

If set $t = 0$, obtain a linear approx to the function $f(x)$ for x in some open ball centered at a, $\mathcal{B}(a,\delta)$.

$$f(a+\delta) \approx f(a) + \delta f'(a).$$

Second order T-expansion

$$f(a+\delta) \approx f(a) + \delta f'(a) + \frac{1}{2}\delta^2 f''(a)b.$$

P-th order T-expansion

$$f(a+\delta) = f(a) + \sum_{i=1}^{p-1} \frac{\delta^i}{i!} f^{(i)}a + \frac{\delta^p}{p!} f^{(p)}(a+t\delta).$$

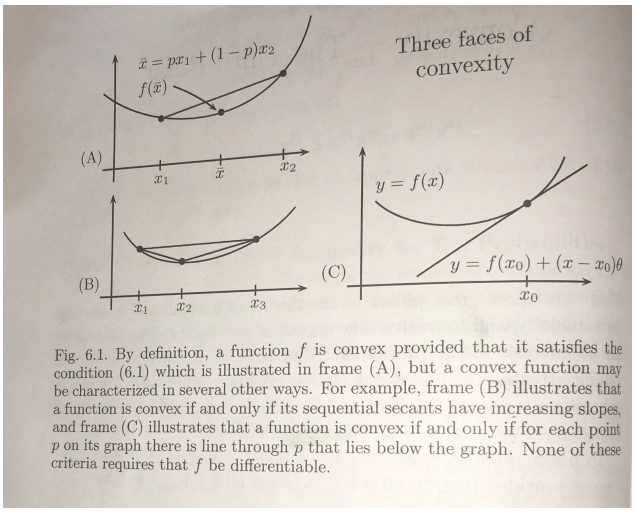
T-expansion regularity condition on $f(x)$ should have a pth derivative continous on $[a, a+\delta]$.

- FACT: Let T-series of $x + \epsilon_n h$ centered at x be $f(x+\epsilon_n h) = f(x) + \epsilon h f'(x) + o(\epsilon_n^2)$. Then for ϵ small the remainder goes away and the following important implication holds

$$h \cdot f(x) > 0 \Rightarrow f(x+\epsilon h) > f(x)$$

2.2.2 Convexity

[19] We could have simply showed this by taking t-expansion around x_0 is $f(x) \approx f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \ldots$, replace x_0 with x and $x \pm h$ instead of x and we get $f(x \pm h) \approx f(x) + f'(x)(h) + \frac{f''(x)}{2!}(h)^2 + \ldots$, yielding T-expansion of $x+h$ around x. Could have done it this way but didn't, c is pretty important since it's a direct connection to mean value theorem. Useful in applications



(a) convexity

(b) game of thrones

Figure 1: Many faces of convexity

convex function Let function $f : [a, b] \to \mathbb{R}$ is said to be convex then forall $x, y \in dom(f)$, with $\lambda \in (0, 1)$ the following are equivalent

1. $f(x\lambda + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$
2. In general (*without smoothness restrictions*) convexity equivalent to

$$(\forall x_0 \in dom(f))(\exists \theta \in \mathbb{R}^{dim(f)})(\forall x \in dom(f))[f(x) \geq f(x_0) + \theta(x - x_0)].$$

   Hence, convexity means for any point x_0 we can find a line with $f(x_0)$ as the intercept, that touches the function at f, at $f(x_0)$ and is below everywhere else. This straight line is called the subgradient.

   - **With smoothness restrictions**

$$(\forall x_0 \in dom(f))(\exists \theta \in \mathbb{R}^{dim(f)})(\forall x \in dom(f))[f(x) \geq f(x_0) + \triangle f(x)^T(x - x_0)].$$

     The first order Taylor expansion at any point, x_0, is a global under estimator of the function. This one we simply use the gradient.

3. In general (*without smoothness restrictions*) convexity equivalent to

$$(\forall a, b \in dom(f)[a < b])(\forall x \in (a, b))[\frac{f(x) - f(a)}{x - a} \leq \frac{f(b) - f(x)}{b - x}].$$

   Hence, convexity means that the sequential secants have increasing slope

   - **With smoothness restrictions**

$$f''(x) \geq 0$$

     The function f has nonnegative curvature everywhere.

Looking at the figure 1, note that convexity condition (1) corresponds to panel A, (2) to C, and (3) to B. Assuming necessary smoothness conditions on f let's prove a couple equivalences:

Proof. (1) $\Rightarrow$ (2) f convex then

$$f(x + \frac{1}{n}(y - x)) \leq f(x) + \frac{1}{n}(f(y) - f(x))$$
$$\iff \frac{f(x + \frac{1}{n}(y - x)) - f(x)}{1/n} \leq (f(y) - f(x)) \text{ take } n \text{ to infinity}$$
$$f(x)'(y - x) + f(x) \leq f(y)$$

The last part follows by *definition of directional derivative* $hf'(x) = \lim_{\epsilon\to\infty} \frac{f(x+\epsilon h)-f(x)}{\epsilon}$

(1) $\Leftarrow$ (2) Let $z = \lambda x + (1-\lambda)y$, then we have by assumption

$$f(y) \geq f(z) + f'(z)(y-z) \iff \lambda f(y) \geq \lambda[f(z) + f'(z)(y-z)]$$

$$f(x) \geq f(z) + f'(z)(x-z) \iff (1-\lambda)f(x) \geq (1-\lambda)[f(z) + f'(z)(x-z)]$$

and by adding the two terms we have

$$\lambda f(y) + (1-\lambda)f(x) \geq f(z) + f'(z)[\lambda y(1-\lambda)x - z] = f(z) = f(\lambda x + (1-\lambda)y)$$

□

Now property (2) is important for maximization. So suppose f is convex and differentiable then any point that satisfies $\triangle f(\bar{x}) = 0$ is a global min. To show this simply substitute $\bar{x}$ into equation 2, and this yields $f(y) \geq f(\bar{x})$, the result follows.

A functions increases the most in the direction of the Jacobian Consider the directional derivative$\langle \triangle f(x_0), h\rangle$ were h is arbitrary direction, by Cauchy Shwartz

$$|\langle \triangle f(x_0), h\rangle| \leq |\triangle f(x_0)||h|.$$

If we let $h = \triangle f(x_0)$ then we have equality, and so the highest value of the direction derivative is in the direction of the gradient. *Thus the gradient is the direction of the function that increases the most. Therefore if we want to take a small step from x_0 to $x_0 + h$, the step that we should go s.t the first order term increases by as much as possible is exactly the gradient. The step that we should go s.t the first order term decreases by as much as possible is exactly the opposite direction of the gradient.* Now if f is convex we use the *global underestimator property*, this means that the Jacobian is a monotonically increasing function. So if say x is not optimal if we take a step in the positive direction of the gradient this makes x even less optimal, thus we should take a step towards the negative direction of the gradient. This intuition is the basis for the gradient descent algorithm that well we will describe next

Subgradient Keeping with the theme of how great property 2 is, we generalize to functions that don;t have a derivative. For a convex and non-differentiable function the sub-gradient is a straight line the touches the convex function and is always below the function

$$f(u) \geq f(w) + \langle \triangle f(w), u - w\rangle$$

. More generally f is convex if for all w in the domain there exists a z s.t

$$f(u) \geq f(w) + \langle z, u - w\rangle$$

. This is not a new definition of convexity, this basically restatement of panel C figure 1. Thus a vector z that satisfies this equation is called a sub-gradient of f at w. The set of subgradients of f at w is denoted by $\partial f(w)$. If f is differentiable at w the the set $\partial f(w)$ contains a single element - the gradient of f at w, $\triangle f(w)$ Convex functions have a subgradient at every point. For the non-smoothcase just think of the absolute value, graph it, and see all the subgradients you can make.

Some other useful properties from convexity:

- monotone gradient condition: $(\triangle f(x) - \triangle f(y))^T(x-y) \geq 0$ follows from property (2)
- *Jensen Inequality:* $f(\sum_{j=1}^n \lambda_j x_j) \leq \sum_{j=1}^n \lambda_j f(x_j)$
- *Strict Convex:* $f(x\lambda + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y)$
- Convexity can proof *AM-GM* inequality: By the differential criterion $f(x) = e^x$ is convex, by jensen $e^{(\sum_{j=1}^n \lambda_j x_j)} \leq \sum_{j=1}^n \lambda_j e^{(x_j)}$ by property of e this is equal to $\prod e^{\lambda_j x_j} \leq \sum \lambda_j e^{(x_j)}$, let $y_j = e^{x_j}$ then we have the AM-GM bound $\prod y_j \leq \sum \lambda_j y_j$

Strict Convexity: Replace the convexity definition w/ strict inequality. Any function that satisfies the strict inequality conditions is strictly convex.

Strong Convexity: A function is strongly convex if f is strictly above the tangent, and so a strongly convex function grows faster than a linear function. For a convex function f, at any point w we can find a linear function that equal to f at w, and does not exceed f at any other point. f is strongly convex if it strictly above the tangent. Strong convexity is important since it is oft used for guaranteeing a linear convergence rate of many gradient decent based algorithms. Note that strong convexity implies strict convexity. So there are no flat parts, put simply a strong convex function is above it's first order approximation. If twice differentiable then $m = \triangle^2 f(w)$.

A function f is m-strongly-convex over $dom(f)$ with respect to norm $||.||$ if for any $w \in dom(f)$ we have

$$(\forall z \in \partial f(w))(\forall u \in dom(f))[f(u) \geq f(w) + \langle z, u - w\rangle + \frac{m}{2}||u - w||^2]$$

Equivalent characterizations.

1. $\triangle^2 f(x) \geq mI$
2. $g(x) = f(x) - \frac{m}{2}||x||^2$is convex for all x
3. $(\triangle f(x) - \triangle f(y))^T (x - y) \geq m||x - y||$
4. $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) - \frac{\lambda(1-\lambda)m}{2}||x - y||^2$

If not differentiable replace condition w/ subgradient!

Boundedness and Lipschitz continuity of convex function Let f be a convex function and let K be a compact set contained in the relative interior of the domain $Domf$ of f. Then f is Lipschitz continuous on K there exists constant L the Lipschitz constant of f on K such that

$$|f(x) - f(y)| \leq L|x - y|, \forall x, y \in K$$

. Relative interior of S is the interior of the affine hull of S

$$\text{aff}(S) = \{\sum \alpha_i x_i | k > 0, x_i \in S, \alpha_i \in R, \sum \alpha_i = 1\}$$

only diff between affine hull and convex hull is that α_i doesn't have to be greater than or equal to 0. This definition is for spaces that live in a lower dimension.

2.2.3 Online Sub-Gradient Descent

The intuition of the sub-gradient algorithm is that if you start from some point and you go negative to direction of the sub-gradient then you are going to decrease the convex loss function.

Online Gradient Descent

input f, T, initial point $\mathbf{x}_1 \in S = \mathbb{R}^N$, η parameter: $\eta > 0$
update rule: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{z}_t, \mathbf{z}_t \in \partial f(\mathbf{w}_t)$

$\mathbf{w}_t \in S$ where S is a compact set. $\mathbf{w}_t$ is the prediction at period t. f is the loss at period t. We going in some small step η. We have that $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{z}_t$, and $\mathbf{w}_{t+1}$ should do better against f than $\mathbf{w}_t$ because we are going in the direction in which f decreases the most. S=U, but we will generalize to the case when $S \subset U$

Now let's look back at property (2) of convexity, *the linear lower bound property of convexity*. This property is the essence of the algorithm we just built so we will go into extreme depth analyzing it.

Linear lower bound characterization of convex function f

For a convex function we can bound it below by a linear function.

$$f(u) \geq f(w) + \langle \triangle f(w), u - w \rangle$$

So basically everything in economics/operations is based on the idea that if you look at the value of a *convex function you can bound it from below, by many linear functions, these linear functions are all the subgradients of all the points.* So you can bound a convex function from below by a line. Look at the beautifully drawn figure below, we see that pictorially that $f(u) \geq f(w) + \langle \triangle f(w), u - w \rangle$, so $f(u)$ bdd from below by a straight line, aka $f(u)$ greater than or equal to the first order taylor approximation centered at w.

We also have the equivalent form of this identity

$$f(w) - f(u) \leq \langle \triangle f(w), w - u \rangle \iff$$
$$f(w) - f(u) \leq \langle -\triangle f(w), u - w \rangle$$

From this we can see that we can *bound the difference of a convex function by looking at which direction we are going* in this case going from w to u and multiplying by the slope
From from property (2) we have the *monotone gradient condition*, this is just by adding the property at w_{t+1} and at w_t

$$(\triangle f(w_{t+1}) - \triangle f(w_t))^T (w_{t+1} - w_t) \geq 0$$

. This condition says that if $x \geq y$ then $\triangle f(w_{t+1}) \geq \triangle f(w_t)$, so the derivative is weakly increasing in its argument. [a]

[a] The reason we are spending so much time on this property of convexity is that (1) we are obviously going to use it for subgradient descent algo (2) This property is used in 'basically every proof' in the Mechanism Design course Eran teaches, so its important to have it etched in stone

Figure 2: Linear Lower Bound of Convexity

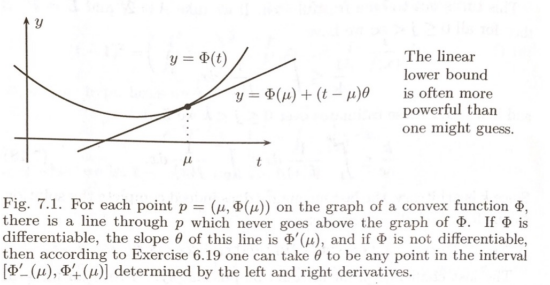
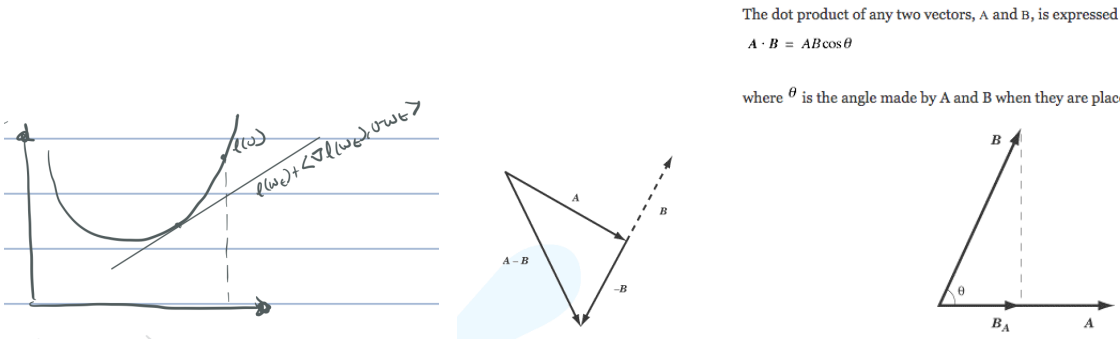


Fig. 7.1. For each point $p = (\mu, \Phi(\mu))$ on the graph of a convex function Φ, there is a line through p which never goes above the graph of Φ. If Φ is differentiable, the slope θ of this line is $\Phi'(\mu)$, and if Φ is not differentiable, then according to Exercise 6.19 one can take θ to be any point in the interval $[\Phi'_-(\mu), \Phi'_+(\mu)]$ determined by the left and right derivatives.

Let's relate what we learned to regret about convexity to regret:

$$f(\mathbf{w}_t) - f(\mathbf{u}) \leq \langle \triangle f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{u} \rangle$$

recall the LHS is the instantenous regret, $r_{i,t}$ (See 1.3.2), which is the regret felt from not follow $\mathbf{u}$ at the end of period t, and we know that $\sum_{k=1}^{T} r_{i,k} = R_{i,T}$. By this bound we see that the RHS is positive if we have regret for period t. So if we suffer regret at the end of period t this means $f(\mathbf{w}_t) > f(\mathbf{u})$, by monotone gradient condition this means $\mathbf{w}_t > \mathbf{u}$. So if we go in direction $-\triangle f(\mathbf{w}_t)$ the next period we get closer to $\mathbf{u}$, and our function gets closer to $f(\mathbf{u})$ as well. So every period if we suffer regret, aka some vector $\mathbf{u}$ is doing better than $\mathbf{w}_t$ then our algorithm will get closer to $\mathbf{u}$ which makes it more difficult to suffer big regret tommorow. So the intuition the subgradient descent algorithm algorithm is as follows either the learner is doing better than $\mathbf{u}$ or the learner is



(a) convex function bdd below by a line (b) vectors and subtraction (c) dot product is 0 when perpendicular, dot product positive if angle small.

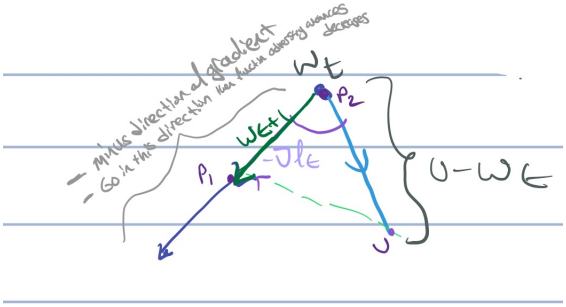
Figure 3: Just a bunch of graphs I found useful

going to get close to $\mathbf{u}$ in the future. *So if we suffer regret at the end of period t then*

$$||\mathbf{u} - \mathbf{w}_{t+1}|| \leq ||\mathbf{u} - \mathbf{w}_t||$$

Now in the proof note that the adversary announcing a straight line is the same as him announce a convex loss function. To see this note that if the adversary announces the sub-gradient, the subgradient and the function f are the same at $\mathbf{w}_t$. This follows since for every $\mathbf{w}$ when $y = f(\mathbf{w}_t)$ by convexity there exist a line that goes through y but is below the graph of f (aka the line is tangent), aka $\exists \mathbf{z}_t$ st $y = f(\mathbf{w}_t) + \langle \mathbf{z}_t, \mathbf{u}_t - \mathbf{w}_t \rangle$. And every other point that the learner could have chosen the loss will be lower, this follows first because of the linear transition equation, $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{z}_t, \mathbf{z}_t \in \partial f(\mathbf{w}_t)$ the learner moves in the direction in which slope decreases, and second because the line $f(\mathbf{u}_t) = f(\mathbf{w}_t) + \langle \mathbf{z}_t, \mathbf{u}_t - \mathbf{w}_t \rangle$ is below the convex function.[20] We can assume that function that the adversary announces is the linear subgradient, and he toggles w/ $\triangle f$ to mess with us. Essentially adversary just announces regret $f(\mathbf{w}_t) - f(\mathbf{u}) \leq \langle \triangle f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{u} \rangle$.

Figure 4: Angles Product Subgradient Algo



At time t we are at w_t the angle is acute and so $\langle \triangle f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{u} \rangle$ is positive. We want to get closer to $\mathbf{u}$, our updating procedure has $\mathbf{w}_{t+1}$ move along the line on the left aka in the direction of $\triangle f(\mathbf{w}_t)$. Thus the dotted green line is distance from u and w_{t+1} this distance is less than dist between $\mathbf{w}_t$ and $\mathbf{u}$, and hence we are closer to $\mathbf{u}$. The closer to $\mathbf{w}_{t+1}$ is to $\mathbf{u}$ the closer our loss $f_{t+1}(\mathbf{w}_{t+1})$ is to $f_{t+1}(\mathbf{u})$. So the intuition of the algorithm is either the learner is doing better than $\mathbf{u}$ or the learner is going to get close to $\mathbf{u}$ in the future.

[20] From figure 3 panel A we see that subgradient is below the loss function, if we move to the negative of the slope we get less loss, this coincidently is our plan. So if we assume what the adversary announces is the linear subgradient, we are still good because all we need is the sub–gradient to know which way to move (By the linear lower bound convexity property)

Theorem: Bound on regret for online subgradient descent

If the online gradient descent algorithm is run on a sequence of convex loss functions then for all $\mathbf{u}$ is

$$Regret_T(\mathbf{u}) \leq \frac{1}{2\eta}||\mathbf{u}||_2^2 + \eta \sum_{i=1}^{T} ||\mathbf{z}_t||_2^2. \tag{9}$$

Where U is convex and compact. Let $\mathcal{Z} = \cup_{\mathbf{w}\in S}\partial f(\mathbf{w})$ be the set of sub-gradients of S. If we have that the sub-gradients are uniformly bounded, $(\forall z \in \mathcal{Z})[|z| \leq L]$, and the squared average is less than the bound squared[a] [b]
$,\frac{1}{T}\sum_{t=1}^{T} z_t^2 \leq L^2$ then

$$R_T(\mathbf{u}) \leq \frac{1}{2\eta}B^2 + T\eta L^2.$$

In particular if $U = \{\mathbf{u} : ||\mathbf{u}|| \leq B\}$ and $\eta = \frac{B}{L\sqrt{2T}}$ then

$$R_T(\mathbf{u}) \leq BL\sqrt{2T}$$

. Note that B can be interpreted as diameter of the set U.

[a] Why is this here? It's because in the online regression case it makes a huge difference from saying that L is the supremum of all the norms of all the data that you get, vs L is just the average of the norms of all the data that you get. So suppose you have one data point that has a very high norm (in the email regression case, it means one exceptionally long email). Then if you take L to be the supremum you will get a poor regret bound, but if you take L to be the average then you get a much better bound.

[b] Subtle note: Instead of $f(\mathbf{w}_t)$ we should have put $f_t(\mathbf{w}_t)$, since regardless of the convex function every step we get closer $\mathbf{u}$. Now you may say 'what if the adversary announced a loss function every period whose derivatives pointed in the opposite direction of the last'. If the adversary does this then we are jumping over $\mathbf{u}$ all the time, but since we are moving in small steps, η, the total loss will be asymptotically close to the loss of playing $\mathbf{u}$ all the time. So η has it uses! (This would be the expert problem when the adversary makes one expert correct in odd periods and one expert correct in even, like a sine function)

First we prove the case where $\mathbf{w}_1 \in U$, $S = \mathbb{R}^N$, $U \subseteq S$ is bdd set, and the learners compares to experts who can only choose points in U. Recall that U the set of experts, S is the set learner chooses from

Proof. We first suppose $\mathbf{w}_1 \in U$ (U the set of experts) , and $S = \mathbb{R}^N$ (S the set learner chooses from) since $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\mathbf{z}_t, \mathbf{z}_t \in \partial f(\mathbf{w}_t)$ we have the following

$$\begin{aligned}
||\mathbf{u} - \mathbf{w}_{t+1}||^2 = ||\mathbf{u} - \mathbf{w}_t + \eta\mathbf{z}_t||^2 &= ||\mathbf{u} - \mathbf{w}_t + \eta\mathbf{z}_t||^2 \\
&= ||\mathbf{u} - \mathbf{w}_t||^2 + 2\eta\langle \mathbf{z}_t, \mathbf{u} - \mathbf{w}_t\rangle + \eta^2||\mathbf{z}_t||^2 \\
2\eta\langle -\mathbf{z}_t, \mathbf{u} - \mathbf{w}_t\rangle &= ||\mathbf{u} - \mathbf{w}_t||^2 - ||\mathbf{u} - \mathbf{w}_{t+1}||^2 + \eta^2||\mathbf{z}_t||^2 \\
\langle -\mathbf{z}_t, \mathbf{u} - \mathbf{w}_t\rangle &= \frac{1}{2\eta}[||\mathbf{u} - \mathbf{w}_t||^2 - ||\mathbf{u} - \mathbf{w}_{t+1}||^2] + \frac{\eta}{2}||\mathbf{z}_t||^2
\end{aligned}$$

From convex argument we have $f(\mathbf{w}_t) - f(\mathbf{u}) \leq \langle -\mathbf{z}_t, \mathbf{u} - \mathbf{w}_t\rangle$ and so we have

$$f(\mathbf{w}_t) - f(\mathbf{u}) \leq \langle -\mathbf{z}_t, \mathbf{u} - \mathbf{w}_t\rangle = \frac{1}{2\eta}[||\mathbf{u} - \mathbf{w}_t||^2 - ||\mathbf{u} - \mathbf{w}_{t+1}||^2] + \frac{\eta}{2}||\mathbf{z}_t||^2 \Rightarrow$$

$$\begin{aligned}
R_T(\mathbf{u}) = \sum_{t=1}^{T} f(\mathbf{w}_t) - f(\mathbf{u}) \leq \sum_{t=1}^{T}\langle -\mathbf{z}_t, \mathbf{u} - \mathbf{w}_t\rangle &= \sum_{t=1}^{T} \frac{1}{2\eta}\underbrace{[||\mathbf{u} - \mathbf{w}_t||^2 - ||\mathbf{u} - \mathbf{w}_{t+1}||^2]}_{\text{telescoping series}} + \sum_{t=1}^{T}\frac{\eta}{2}\underbrace{||\mathbf{z}_t||^2}_{\leq L} \\
&\leq \frac{1}{2\eta}||\mathbf{u} - \mathbf{w}_1||^2 + \sum_{t=1}^{T}\frac{\eta}{2}(L)^2 \\
&\leq \frac{1}{2\eta}[B]^2 + \frac{T\eta}{2}(L)^2
\end{aligned}$$

Thus $R_T(\mathbf{u}) \leq \frac{1}{2\eta}B^2 + \frac{T\eta}{2}L^2 \leq \frac{1}{2\eta}B^2 + T\eta L^2$, so if $\eta = \frac{B}{L\sqrt{2T}}$ then RHS is equal to $BL\sqrt{2T}$ □

Now suppose we want to restrict ourselves to $S \subsetneq \mathbb{R}^n$, where $U \subset S$.[21] So S (the set the learner chooses from) is not $\mathbb{R}^N$ but S still contains U. Note that in the previous algorithm/proof we had $S = \mathbb{R}^n$, for our choices $\mathbf{w}_t \in S$ we didnt have to make sure the choice was in set S since S was $\mathbb{R}^n \ldots$ the whole space. Before we begin let's define a couple of terms

Projection onto a convex set The projection operation onto a convex set is defined as the closest point inside the convex set to a given point

$$\Pi_S(\mathbf{y}) := \arg\min_{\mathbf{x}\in S} ||\mathbf{x} - \mathbf{y}||).$$

The projection of a given point over a compact convex set exists and is unique [a]

Pythagorus[b] Let $S \subset \mathbb{R}^d$ be a convex set, $\mathbf{y} \in \mathbb{R}^d$ and $\mathbf{x} = \Pi_S(\mathbf{y})$ then for any $\mathbf{z} \in S$ we have

$$||\mathbf{y} - \mathbf{z}|| \geq ||\mathbf{x} - \mathbf{z}||$$

[a] recall separating hyperplane proof from 560-1

[b] So by Seperating Hyperplane we can seperate $\mathbf{w}_{t+1}$ and the set S w/ a hyperplane that goes through the projection point. $\mathbf{w}_{t+1}$ is perpendicular to hyperplane so makes a rectangular angle w/ it (90 degrees). Connect all the points to be a triangle the hypotenuse angle is more than rectangle thus ..... tldr rectangular angle. Just remember this argument in relation to triangles

Whoops forgot the gradient descent algorithm changes slighly as well

Online Sub-Gradient Descent (v2)

This is the case when $S \neq \mathbb{R}^N$, aka $S \subsetneq \mathbb{R}^n$ and again $U \subseteq S$

input f, T, initial point $\mathbf{x}_1 \in U$, η
parameter: $\eta > 0$
update rule: $\mathbf{x}_{t+1} = \mathbf{w}_t - \eta\mathbf{z}_t, \mathbf{z}_t \in \partial f(\mathbf{w}_t)$, and $\mathbf{w}_{t+1} = \Pi_S(\mathbf{x_{t+1}})$

Same algo, just pick closest point in constraint set.

Now we proof the previous theorem (9).

Proof. Note that if $\mathbf{x}_{t+1} \notin S$ then for $\mathbf{w}_{t+1} = \Pi_S(\mathbf{x_{t+1}})$ we have

$$||\mathbf{x}_{t+1} - \mathbf{u}|| \geq ||\mathbf{w}_{t+1} - \mathbf{u}||.$$

but we have

$$||\mathbf{u} - \mathbf{w}_t + \eta\triangle f(\mathbf{w}_t)|| = ||\mathbf{x}_{t+1} - \mathbf{u}|| \geq ||\mathbf{w}_{t+1} - \mathbf{u}||.$$

So in the previous proof if we switch the first equality to an inequality we have out result, since

$$||\mathbf{w}_{t+1} - \mathbf{u}|| \leq ||\mathbf{u} - \mathbf{w}_t + \eta\triangle f(\mathbf{w}_t)||$$

where $\mathbf{z}_t = \triangle f(\mathbf{w}_t)$ I'm just being lazy w/ notation □

Exercise 2 Find the best η via calculus

[21] If $S \subset U$ we would never get a no regret strategy. Why ... hint: it has to do w/ adversarial environment

2.3 Application: Using regret bounds for Gradient Descent and Exponential Weighted Avg

Application of bounds: Expert Problem $U = S = \triangle(N)$
$l_t = \langle \mathbf{w}_t, \mathbf{y}_t \rangle$, $0 \leq |l_t| \leq 1$

Now to use our gradient descent bound we need to find B and L

Recall that B is the diameter of the set U. To find the distance consider the farthest points in the simplex, this would be the extreme points. So take two extreme points $x = (1, 0, 0, \ldots, 0)$ and $x' = (0, 1, 0, 0, \ldots, 0)$, the euclidean norm of these would be $\sqrt{2}$, and so $B = \sqrt{2}$

Recall that L is th bound on the sub-gradients of our loss functions, $\frac{\partial l_t}{\partial \mathbf{w}} = \mathbf{y_t}$, where $\mathbf{y}_t \in \{0, 1\}^N$, thus $||\mathbf{y}_t|| \leq \sqrt{N}$, and so $L = \sqrt{N}$

Bound for regret via subgradient descent if we set $\eta = \frac{B}{L\sqrt{2T}} = \frac{1}{\sqrt{NT}}$: $\sqrt{2}\sqrt{N}\sqrt{2T} = 2\sqrt{NT} = O(\sqrt{NT})$
Bound for regret via rando-exp-weight-avg descent: $O(\sqrt{\log N \cdot T})$
So rando-exp-weight-avg has a better bound on regret.

The reason the expert algorithm is better than subgradient descent in this context, is that the expert algorithm is on the simplex and not built for norm l_2, it does much better with norm l_1 and norm l_∞. Since the simplex is essentially the unit ball of the norm l_1, and the regret bound says something about the l_∞ bound.

Exercise 3 Go to Shai Shalev-Shwartz book and look at the more general algorithm which contains both subgradient descent and randomized exponential weighted average (REWA), by choosing different parameters, and explains better the relationship between the two algorithms.

Note that the proof of both algorithms have some structure that is similar, in REWA the main intuition proof was 'every period either we are doing well or expert suffer some penalty,' in subgradient descent the main intuition in proof was 'every period either we are doing well or we get closer to $\mathbf{u}$, the optimal solution'

Exercise 4 Show that under the conditions of Thm (9) if learner is trying to optimize over the unit ball in R^N then

$$Regret_T(u) \geq O(\sqrt{NT})$$

However there are times when we can get better than $O(\sqrt{NT})$, this is related to the loss function being strongly convex. Before we go into this we define a lemma associated w/ strongly convex functions.

S be non empty convex set. Let f where $dom(f) = S$ be a σ-strong convex function over S w/ respect to norm $||.||$. let $u = \arg\min_{v \in S} f(s)$ aka the minimum of the function. Then, for all $w \in S$

$$f(w) - f(u) \geq \frac{\sigma}{2}||w - u||^2$$

Many functions we deal with that are convex and do not have linear components are σ-strongly convex. [a]

[a] Such as the quadratic loss function. e^x strong convex if restricted to compact set, just use second derivative def of strong convexity to show this.

The intuitive proof is to assume f is differentiable and w is in the interior, let $f^* = f(w)$, then FOC means $\triangle f^* = 0$, using the definition of strong convexity to eliminate we get the result.

Now suppose w is on the boundary of S then if $\langle \triangle f(w), u - w \rangle < 0$ then since a σ-convex function is convex, we have $f(u) - f(w) \geq \langle \triangle f(w), u - w \rangle < 0$ and so $f(w) > f(u)$ contradiction since $f(w)$ is the min. Alternatively

if $\langle -\triangle f(w), w-u\rangle < 0$, since $\langle -\triangle f(w), w-u\rangle$ is the directional derivative in direction $w-u$, by definition of directional derivative since it is negative if we take ϵ step from $f(w)$ in direction of $w-u$ we decrease function and so $f(w)$ can not be the min.

Figure 5: σ- convex

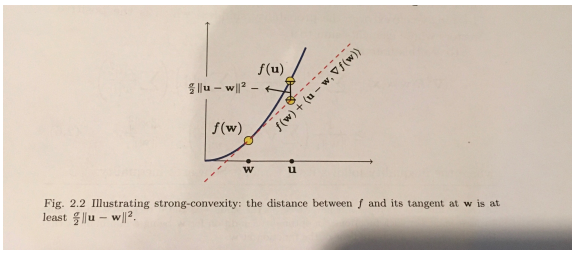


Fig. 2.2 Illustrating strong-convexity: the distance between f and its tangent at $\mathbf{w}$ is at least $\frac{\sigma}{2}\|\mathbf{u}-\mathbf{w}\|^2$.

Now since strong convexity implies strict convexity we know that the min value is unique. Now suppose we have a σ-strong convex loss function, and u is optimal value. If w is far from u then we regret a large amount since $\frac{\sigma}{2}||u-w||^2$ is quadratic, and reget $f(w)-f(u)$ is greater than the quadratic term. Also the subgradient will be large as well.

Exercise 5 Show that if the learner is trying to optimize the unit ball in R^N under the condition of Thm (9), and the loss function is strong convex then

$$Regret_T(u) \geq O(\log T)$$

where the $\log T$ depends on how much the loss function is strongly convex.

Online Regression w/ Absolute Loss

We choose the online regression with absolute value because it is more common when working w/ large data and sparse regression [a], also since the ols w/ square distance is strongly convex so we get much better bounds then the bound we spent all this time proving.
To use Thm (9) we assume $||\mathbf{u}||_2 \leq B$ so essentially assuming that U is a ball of radius B with respect to the l_2 norm. [b] To use our theorem S must be convex/compact.

for $1, 2, \ldots, T$
choose $\mathbf{w}_t \in S$
adversary announce data (x_t, y_t)
compute loss: $|y_t - \langle \mathbf{w}_t, \mathbf{x}_t \rangle|_1$
compute transition: $\mathbf{w}_{t+1}$

For the GD-algo we need update to be $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{z}_t, \mathbf{z}_t \in \partial f(\mathbf{w}_t)$. From our loss function we have that

$$f(\mathbf{w}_t) = |y_t - \langle \mathbf{w}_t, \mathbf{x}_t \rangle| = \begin{cases} y_t - \langle \mathbf{w}_t, \mathbf{x}_t \rangle & \text{if } y_t > \langle \mathbf{w}_t, \mathbf{x}_t \rangle \\ y_t + \langle \mathbf{w}_t, \mathbf{x}_t \rangle & \text{o.w} \end{cases}$$

In case 1, $\frac{\partial f(\mathbf{w}_t)}{\mathbf{w}_t} = -\mathbf{x}_t$, in case two, $\frac{\partial f(\mathbf{w}_t)}{\mathbf{w}_t} = \mathbf{x}_t$, and so the updating procedure is

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + \eta x_t & \text{if } y_t > \langle \mathbf{w}_t, \mathbf{x}_t \rangle \\ \mathbf{w}_t - \eta x_t & \text{o.w} \end{cases}$$

In this case $B = B$, and $L = |||\mathbf{x}_t|||_\infty$ aka the supremum of all the l_1 norms [c].

[a] please explain why
[b] Note that if there are not restrictions on the coefficients than the adversary can cause unbdd regret, since he can make it that coefficients that are larger and larger are better than u.....this is not very clear.
[c] by Thm (9) instead of L, we could have but the bound as the average of the gradients. (refer to the thm if confused about this statement) In online regression think of $\mathbf{x}_t$ as a very sparse vector whose dimension can be very large, so in principle the norm of $\mathbf{x}_t$ can be $\sqrt{dim(\mathbf{x}_t)}$. In the spam filtering case (section 2.1.2) $\mathbf{x}_t$ is the number of words in dictionary again very sparse, and so the norm of $\mathbf{x}_t$ would be the average length of a email. In this case and in the regression case in the assumption on environment has to be restricted s.t the average length of email is less than some value. This can be done via law of large numbers, aka suppose emails are i.i.d.

2.4 Let me go on a monologue

To conclude we showed how to use Thm (9), we showed what should be the updating rule under GD, and we explained the meanings of the bounds L, B in terms of the problem the theorem.

Exercise 6: Do the same exercise for the portfolio problem.